Internal **Technical** Education

http://mste

# ECS Talk:
# The future evolution of Windows Engineering

**Dave Probert**, Development Lead, Windows Kernel

# Agenda

**Fat but not happy**

Is this anyway to build a house?

Customers!?

Engineers not programmers

From lifestyle to lifecycle

Bottom-up in a top-down company

Summary of challenges

# Windows

**Windows is the best business in the world**

- We have incredibly talented people
- We are leaders in software innovation
- We make a lot of money
- We are fulfilling the vision of empowering ordinary people

# Windows

**Windows is the best business in the world**

- We have incredibly talented people
- We are leaders in software innovation
- We make a lot of money
- We are fulfilling the vision of empowering ordinary people

**But we are victims of our own success**

- It is "the world against Microsoft"
- Monetization of software innovation under attack
- Growth is hard to maintain
- Customer expectations are higher than ever

# Windows is scaling
# Windows engineering is not

Windows is too complicated

Development is disconnected from customers

Our quality mistakes are too costly

Our product development processes are inadequate

# Windows is scaling
# Windows engineering is not

Windows is too complicated

**ARCHITECTURE**

Development is disconnected from customers

**SCENARIO-DRIVEN**

Our quality mistakes are too costly

**TOOLS**

Our product development processes are inadequate

**FORMALIZE**

# Agenda

- Fat but not happy
- **Is this anyway to build a house?**
- Customers!?
- Engineers not programmers
- From lifestyle to lifecycle
- Bottom-up in a top-down company
- Summary of challenges

# The Winchester Mystery House

# The Winchester Mystery House

# WinchesterMysteryHouse.com/story.html

In 1884, a wealthy widow named Sarah L. Winchester began a construction project of such magnitude that it was to occupy the lives of carpenters and craftsmen until her death thirty-eight years later. The Victorian mansion, designed and built by the Winchester Rifle heiress, is filled with so many unexplained oddities, that it has come to be known as the Winchester Mystery House. Sarah Winchester built a home that is an architectural marvel. Unlike most homes of its era, this 160-room Victorian mansion had modern heating and sewer systems, gas lights that operated by pressing a button, three working elevators, and 47 fireplaces. From rambling roofs and exquisite hand inlaid parquet floors to the gold and silver chandeliers and Tiffany art glass windows, you will be impressed by the staggering amount of creativity, energy, and expense poured into each and every detail.

These stairs that lead to the ceiling are just one of the many bizarre features that Mrs. Winchester designed and had built.

## Amazing Facts

- Number of rooms: 160
- Number of stories: prior to 1906 seven; presently four
- Number of acres: originally 162; presently 4
- Number of windows: frames 1,257

# Why did she keep adding on?

"Some say she thought building the house would prevent her death or appease the spirits of the many killed by the Winchester rifles that built her fortune.

http://gocalifornia.about.com

# Architecture

**Encarta:** The design, structure, and behavior of a computer system, microprocessor, or system program, including the characteristics of individual components and how they interact.

# Architecture Keys

- Understand the problem you are solving
- Devise solution as components
- Identify changes to existing components
- Respect layering and component boundaries
- Fear complexity and dependencies
- Understand the system before you change it
- Be sure your scenarios are sufficient
- Review before you commit



# Binary Components

Atomic unit of distribution, servicing & binding characteristics
- Has resources (file, registry, etc)
- Components can contains multiple files and/or registry entries
- Depends on other components & features
- Exposes configurable properties

Component Manifest XML includes extensions for:
- Settings Schema and Default Values (WMI.Config)
- Privacy information
- Event information (Crimson)
- Perf counters/probes (WMI.Monitoring, Perfcounter team)
- Monitoring rules (WMI.Monitoring)
- Security information (CLR, SAFER, Windows Security)
- Shell Tasks/File extensions etc. (Shell team)
- Web-deployment information (Fusion/ClickOnce team)
- MUI information
- Natural User Interface/natural language query (NUI team)
- Game information (Games team)

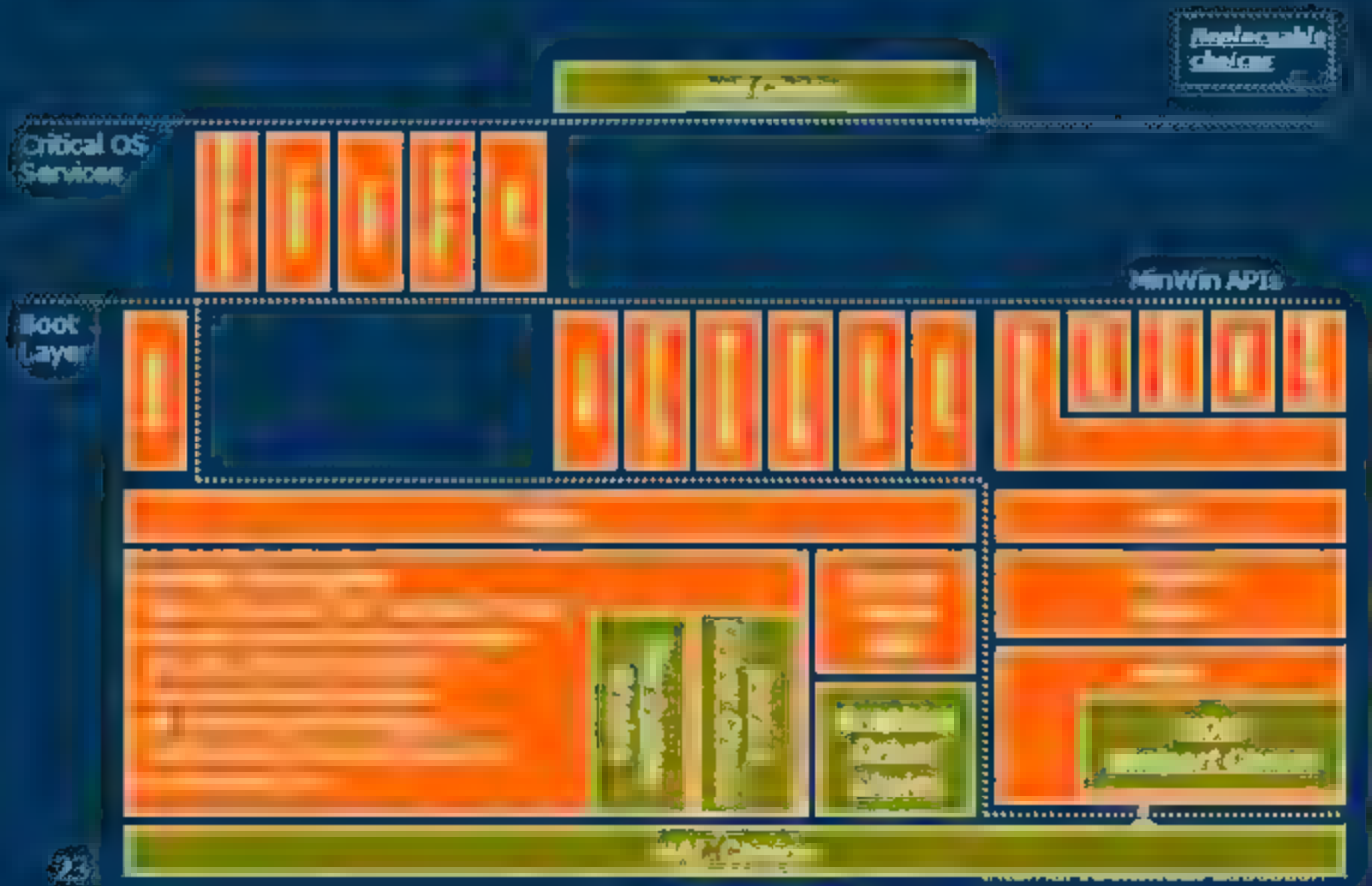
# Source-level Componentization

- **Component:** logical collection of functions/features that work together using internal state or common internal functions
  - Well-defined, completely specified interfaces, protocols, and dependencies
  - Behavior defined by test code
- Source-level components are not binary modules

# Minimal Windows
# The Base of Componentization

- MinWin is the core OS functionality common across all Windows platforms with minimum hardware requirements.
- Windows products built by adding components to MinWin.
- Base testing platform.
- Features:
  - Footprint: 32MB Storage, 64MB RAM
  - Kernel: PnP, Power Management, WMI
    - preserve investments made in device drivers
  - Basic Win32 API: Kernel32.dll, User32.dll, GDI32.dll, subset of Advapi32.dll (registry, service mgmt.)
  - Basic Networking: WinSock 2, TCP/IP, DNS client, DHCP client

# MinWin Architecture



# System Layering

# Windows vs Linux

- Linux said to be more agile:
  - Windows
    - Developer orientation: extending/exploiting the platform, dependencies are good (code reuse, working set), any required components will be installed
    - Development process: centralized, large common platform (Windows, LAPI)
    - Distribution method: small variety of deployments (SKUs), limited installer flexibility, limited out-of-band distribution, monolithic
  - Linux (Open Source movement)
    - Developer orientation : dependencies are potential blockers to use, minimize dependencies to maximize installation and use.
    - Development process: massively distributed, small common platform (Linus' kernel)
    - Distribution method : large variety of deployments, much installer flexibility, no centralized roadmap, componentized

# Agenda

- Fat but not happy
- Is this anyway to build a house?
- **Customers!?**
- Engineers not programmers
- From lifestyle to lifecycle
- Bottom-up in a top-down company
- Summary of challenges

# Deep in the yogurt

- Customer satisfaction is amazingly low
  - Vision has raced ahead of quality
  - We don't build complete solutions
  - We blame the customer
  - Success raises expectations
- We are moving up the food-chain
  - Server customers expect solid engineering
  - The PC-centric lifestyle raises the stakes
- Victims of our own success
  - We are the target
  - The PC Ecosystem is uncontrollable
  - Our complexity is unmanaged

# Jim Allchin: December 1996

It is everyone's responsibility to think about simplifying our software and improving its quality.

Having just spent dedicated time talking to customers, it is clear that this is our single biggest area for improvement.

Regardless of what product you work on or you specific discipline, you need to raise your standards on what is acceptable for customers.

Adding more and more features is not the answer.

Whether at home or at work, people are frustrated with our software.

It is too hard to use and too fragile.

We own changing this.

# Customer connection for developers

- Experience the system as customers experience it
  - **Install software at home**
  - **Spend holidays helping friends & family**
  - **Use the same diagnosis/support tools as PSS**
- Overcome the 99/1 rule
  - **Engineers are geeks! Normal people don't care.**
- Talk to customers

# Linux learnings

What we get so far

- **Importance of community**
- **Advantages of componentization and composability**
- **Open source (sort-of)**

What we still need to get

- **Distributed, darwinian development process**
- **Resonance with systems programmers**
- **Transparency**

# Winning over systems programmers

Why I dislike Windows

- The Win32 API is poorly designed
- Windows is opaque
- Windows is inherently unstable and complex
- Windows doesn't get manageability
- Windows is GUI-oriented
- Windows licensing is PC-oriented
- Windows has an impoverished heritage
- We don't (can't) use Windows as our customers use it

# Winning over systems programmers

What I like about Windows

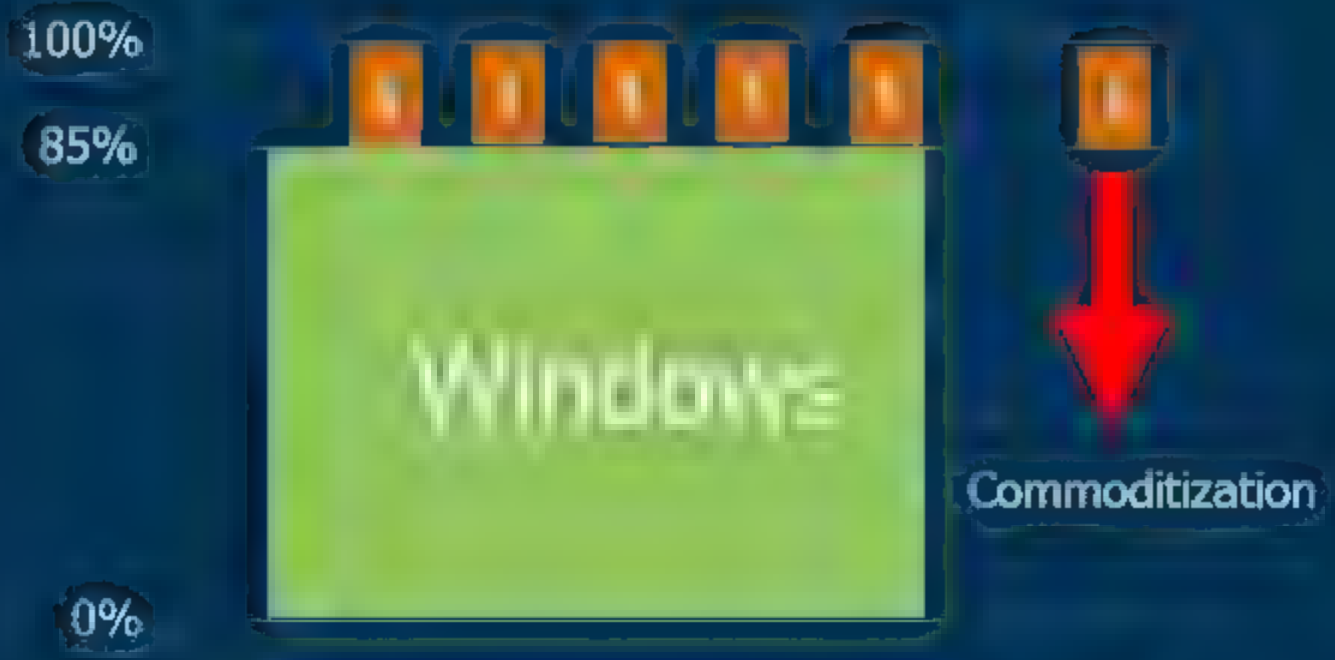
- The NT core is sophisticated and powerful
- The system is extensible (without source)
- NT is *not* unix
- The vision of empowering ordinary people
- Hundreds of millions of customers

# Winning over systems programmers

Why?

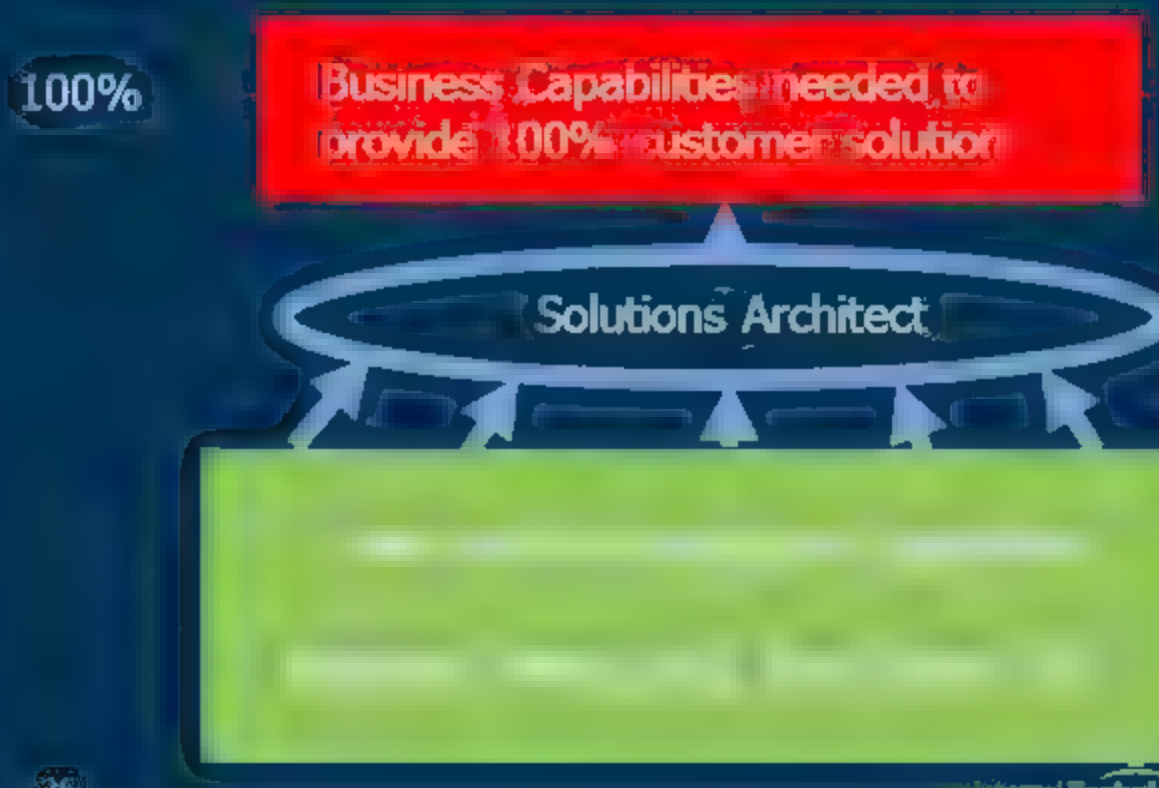


# The 85% problem



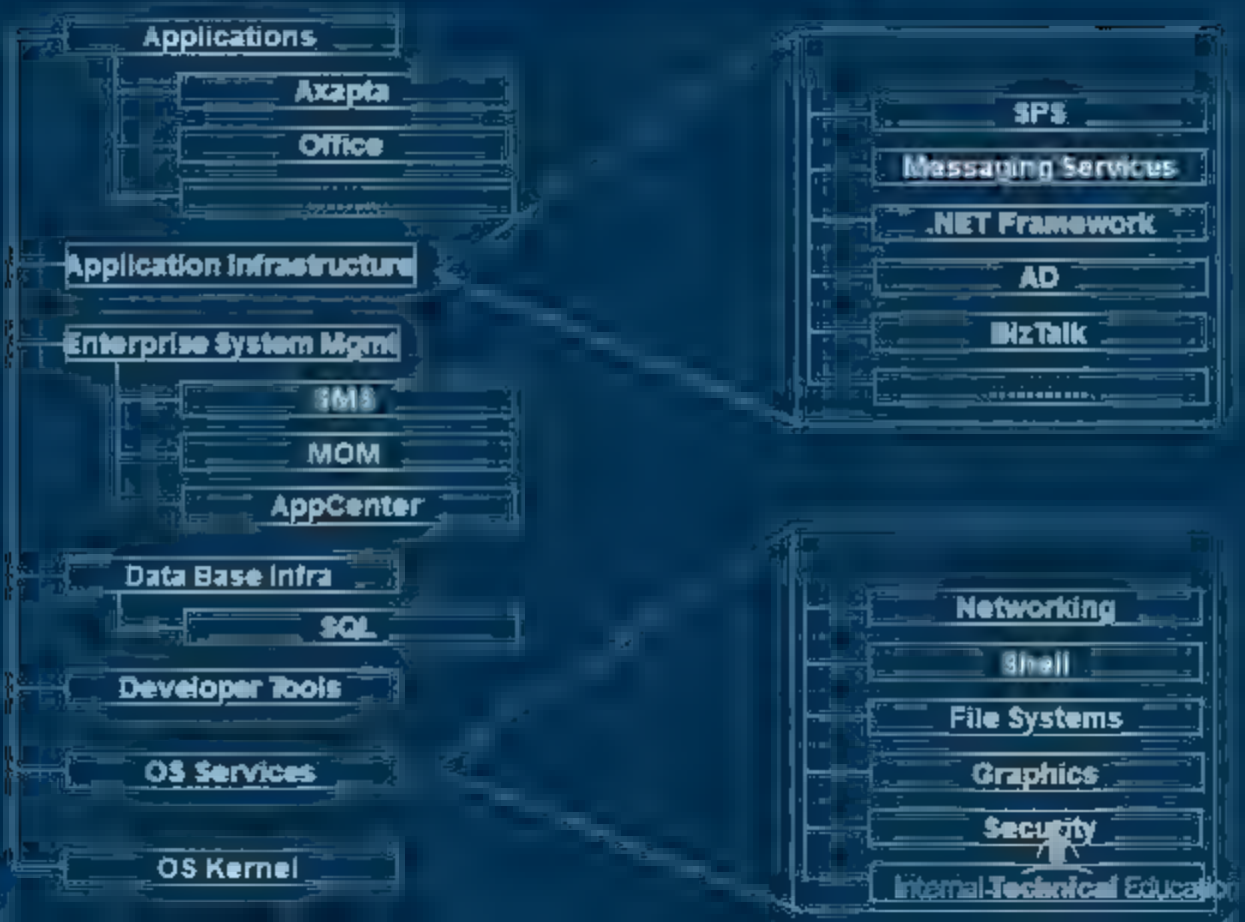
# The Technology Map project

http://MapsServ



# T-Map: Data I/O Model

# T-Map: Current Taxonomy



# Agenda

- Fat but not happy
- Is this anyway to build a house?
- Customers!?
- **Engineers not programmers**
- From lifestyle to lifecycle
- Bottom-up in a top-down company
- Summary of challenges



# What is the difference between a programmer and an engineer?

Engineers run closed-loop

- Measure effectiveness
- Improve processes
- Always learning



# What is the difference between a programmer and an engineer?

- Engineers run closed-loop
  - Measure effectiveness
  - Improve processes
  - Always learning
- Programmers run open-loop
  - Answer is always: *write more code*
  - History doesn't matter
  - Future doesn't matter much either
- If Windows was a bridge...

# The solidification of Windows

- Windows 2000
  - Solid against crashes (Windows components)
- Windows XP
  - Compatibility
- Windows Server 2003
  - Solid against attacks
- Longhorn
  - Solid against 3rd party drivers
  - Solid against hangs
  - Less fragile

# Quality through Tools

- Customer connection (OCA, Watson, ATR, reliability)
- PRE*fix*, PRE*fast*
- Driver Verifier, App Verifier
- Diagnosis (event tracing, Crimson)
- Test automation (code coverage, SelectTest, Archon, WCIT)
- Magellan (Echelon, Injector, Sword)
- Device Driver simulator (behavioral modeling)
- Process automation (SWT)
- Common test criteria
- Monad


Trustworthy Computing
Security
Privacy
Reliability
Business Integrity

# Agenda

- Fat but not happy
- Is this anyway to build a house?
- Customers!?
- Engineers not programmers
- **From lifestyle to lifecycle**
- Bottom-up in a top-down company
- Summary of challenges



# Team Size (dev/test)

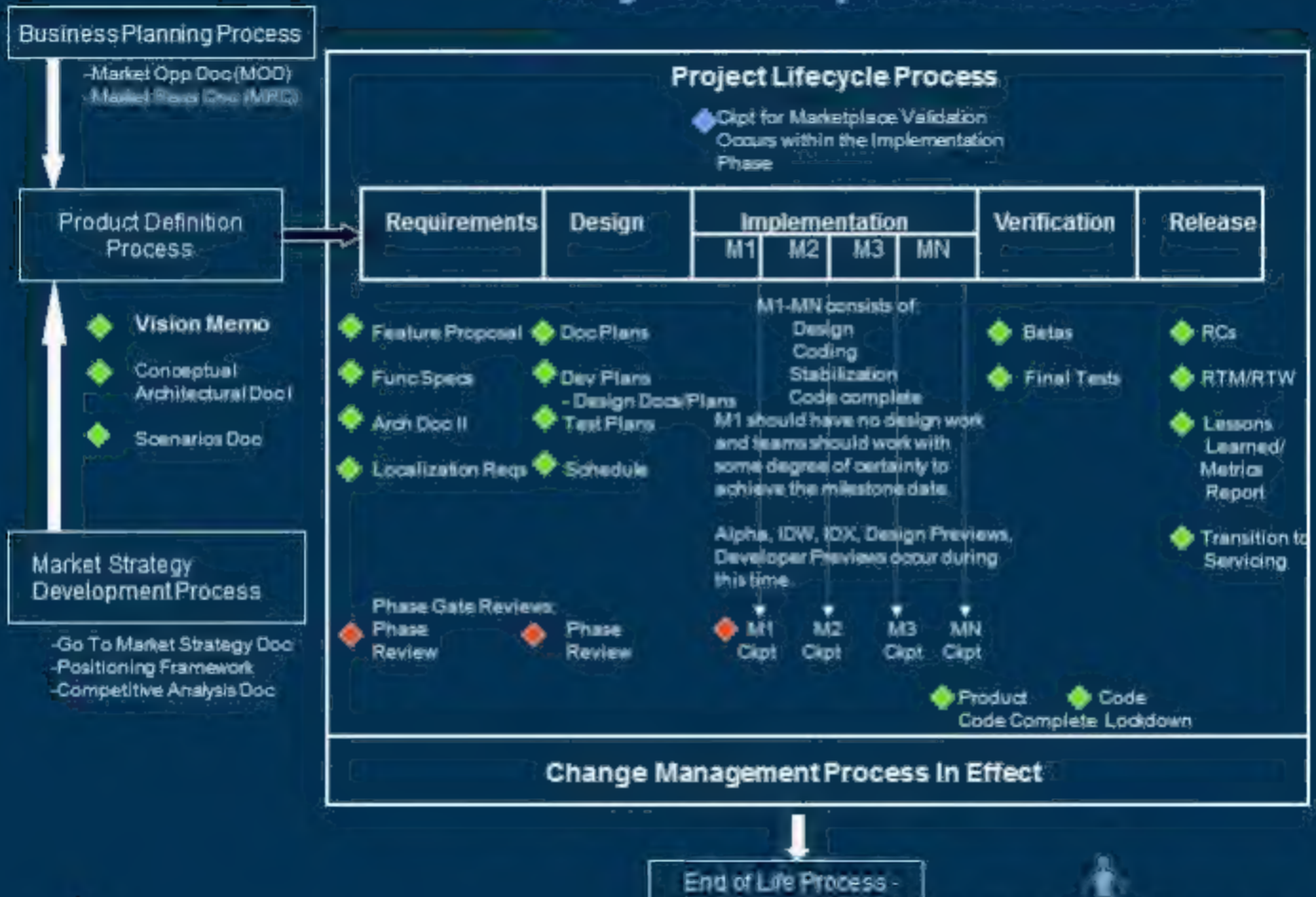
| Product | Dev Team Size | Test Team Size |
|---|---|---|
| NT 3.1 | 200 | 140 |
| NT 3.5 | 300 | 230 |
| NT 3.51 | 450 | 325 |
| NT 4.0 | 800 | 700 |
| Win2k | 1400 | 1700 |
| TODAY | 2800 | 2200 |

# Product Development Process

# Project Lifecycle Guidelines

- The project lifecycle guides the team throughout the development of the project through consistent definitions and milestones/phase gates for better communication. It does not define the methodology/approach to achieve the phase gate/deliverable.
- The definition of a project is that it has a definite beginning and a definite end. There should be CLEAR entry and exit points to each phase.
- There should be no more than 4 or 5 phases to a lifecycle. The lifecycle should be simple enough to be a team's mantra.
- "Planning" is generally not used for a lifecycle phase. Planning should occur throughout the lifecycle as deliverables are continually revisited.
- Words like "ongoing" as the last phase is generally not recommended as it does not indicate an end point to the project.

# Project Lifecycle Framework



# Agenda

Fat but not happy

Is this anyway to build a house?

Customers!?

Engineers not programmers

From lifestyle to lifecycle

**Bottom-up in a top-down company**

Summary of challenges

# *Make it so!* Companies

- Customer is king
  - Inherently top-down
- Vision rules
  - What you can imagine drives what you attempt to do
- Examples
  - Apple computer
  - Ford/GM/Chrysler
  - Microsoft

## Challenge for Windows?

Bottom-up engineering
in a
top-down company